Tunneling Async Data

Document ID: 12991

Contents

Introduction

This sample configuration describes the process of tunneling asynchronous data.

Prerequisites

Requirements

There are no specific prerequisites for this document.

Components Used

This document is not restricted to specific software and hardware versions.

The information presented in this document was created from devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. If you are working in a live network, ensure that you understand the potential impact of any command before using it.

Conventions

For more information on document conventions, see the Cisco Technical Tips Conventions.

Configure

As an example, assume that async RS−232 devices are to be connected by leased line modems. Instead, the leased line modems are replaced with Cisco comm servers. Plug the RS−232 devices into async lines on the Cisco comm servers and connect the comm servers through an arbitrary topology IP network.

In this sample configuration, one side is the caller and the other side is the called. It is presumed that the caller side is more persistent in trying to send data.
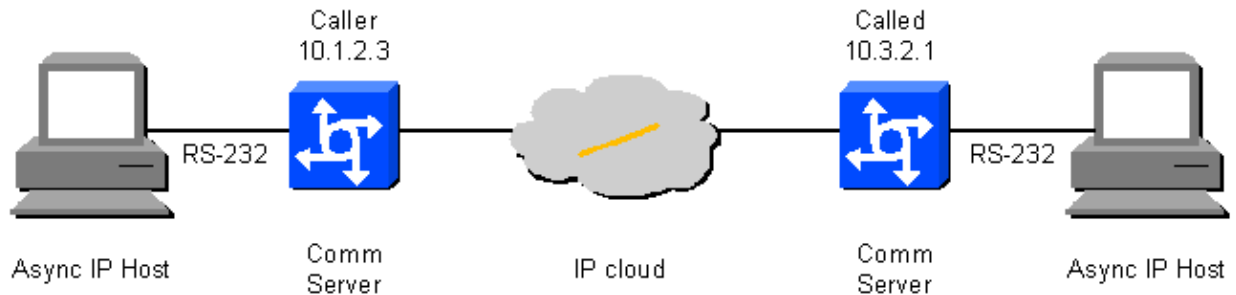
Assumptions:

- The caller side has an IP address of 10.1.2.3 and is using Line 2.
- The called side has an IP address of 10.3.2.1 and is using Line 3.

Note: To find additional information on the commands used in this document, use the Command Lookup Tool (registered customers only) .

Network Diagram

This document uses the network setup shown in the diagram below.

Configurations

This document uses the configurations shown below.

- Caller Side
- Called Side

Caller Side

```
!--- On caller box - 10.1.2.3 define an IP hostname to use on the TELNET so
we can use BUSY-MESSAGE to shut up TELNET.

ip host CALLED-LINE 4003 10.3.2.1          ! port 40xx is raw TCP

!--- Busy-message cannot have a null string - single space works.

busy-message CALLED-LINE \ \               [1]
service tcp-keepalives-out                 [3]
!
line 2
!--- Shut up everything.

no motd-banner

!--- Not available in all versions.

no exec-banner
no vacant-message
autocommand telnet CALLED-LINE  /stream
autohangup

!--- The following command means incoming serial data is saved
until the TCP connection is made.


!

no flush-at-activation
```

```
!--- Not available in all feature sets.

no activation-character

!--- Any character will create the EXEC.

escape-character NONE

!--- This can also be escape-character BREAK.


exec

!--- Need an EXEC to do the TELNET.

special-character-bits 8
exec-timeout 0 0
session-timeout 0 0

!--- RS232 configuration:

no modem inout

!--- Disable modem control [2].


no autobaud
speed 9600

!--- Set the desired speed.

stopbits 1

!--- Alternatively, this can be 2, as desired.

flowcontrol NONE

!--- Alternatively, this can be HARDWARE, or SOFTWARE.

transport input NONE

!--- Do not allow reverse connections.
```

Called Side
| ```
!--- On called box – 10.3.2.1.

no banner incoming
service tcp-keepalives-in            [3]
line 3
no exec
no exec-banner
no vacant-message

!--- RS232 configuration:

modem DTR-active

!--- DTR indicates the status of the TCP connection.

no autobaud
speed 2400
``` |

```
!--- As desired. This does not need to match the speed on the called side.

stopbits 1

!--- Alternatively, this can be 2, as desired.

flowcontrol NONE

!--- Alternatively, this can be HARDWARE, or SOFTWARE.

transport input telnet

!--- Allow the incoming TCP connection.
```

Notes

[1] Unfortunately, it is not possible to specify a null **busy−message** command. It appears that the minimal busy−message is a single space. This means that, if the calling side is unable to establish the TCP connection to the called side, the calling device will send a <CR><LF><space> sequence from the calling RS−232 line (once for each outbound connection attempt). If the **flush−at−activation** command is in effect, then there will be one <CR><LF><space> sequence for each character sent by the calling RS−232 device. If the **no flush−at−activation** command is in effect, then the device will loop, sending <CR><LF><space> sequences until the TCP connection can be established. With the **no flush−at−activation** command, the device is persistent in getting the unsolicited data through.

[2] Use the **no modem inout** command on the calling side. With modem signaling, if the device sees a data set ready (DSR) increase, it will initiate the autocommand. However, if the device is power cycled, and if DSR is high when the device comes up, the autocommand will not be initiated until a **clear line** command is initiated.

[3] Ensure that TCP keepalives are enabled on both sides for the connection of interest; otherwise, if the caller side (or the network path) goes down, the called side will be unaware (unless it has application data to send) that the caller's connection has been dropped causing the new caller side connection attempt to fail.

Verify

There is currently no verification procedure available for this configuration.

Troubleshoot

This section provides information you can use to troubleshoot your configuration.

The following debugs will verify that the lines are coming up and going down, and that the TCP session is starting and stopping:

```
configure terminal
  service timestamp debug date msec
  end
debug modem
debug ip tcp packet N

!--- Where N is the line of interest.
```

If it seems that the async tunneling is failing transparently to pass data, then attach an RS–232 datascope to both async lines and an IP sniffer to the IP path in the middle.

Related Information

- **Dial Technology Support Pages**
- **Technical Support – Cisco Systems**

Updated: Nov 19, 2007   Document ID: 12991